Raw image width

image   pixel

101

Window base address

Window width = n * 8 pixels

line 1

Line increment = m
(m = number of lines)

121

DMA

interrupt_sliding_window_ready   125

the DMA prepares
the next window and
issues an interrupt
when it is loaded

base address

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F

8 pixels wide

left

Sliding Window 1

right

Sliding Window 2

8 pixels high

next Sliding Window

111   113   115

131

x_bus

iniDSP

iniDSP has access
to the first 2 sliding
windows

FIG. 1

image

1st	2nd	3rd
4th		

filling direction of sliding window

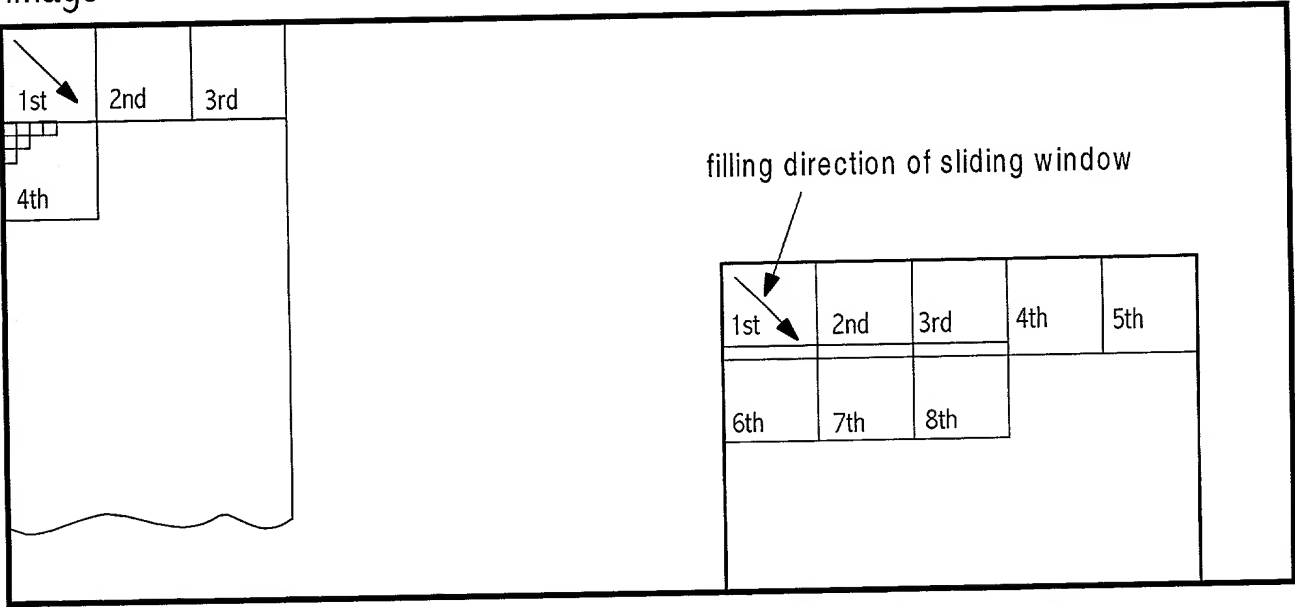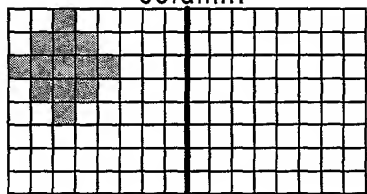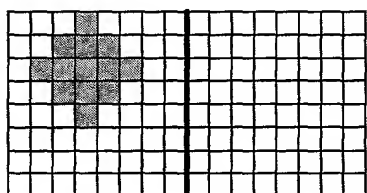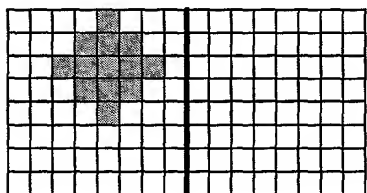| 1st	2nd	3rd	4th	5th
6th	7th	8th		

FIG. 2

Initial condition, 5x5 filter will be
processed centered on the third row,
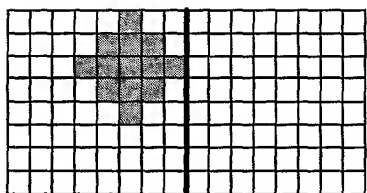starting centered on the third
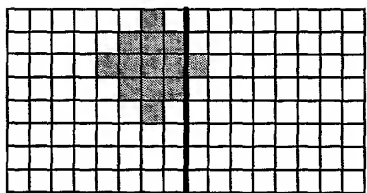column.

Filter centered on 4th
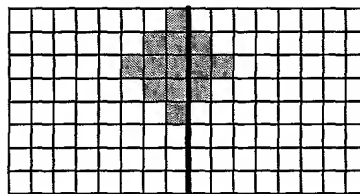column

Filter centered on 5th
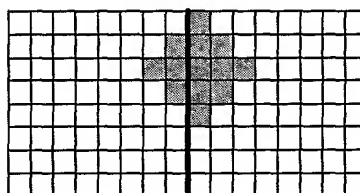column

Filter centered on 6th
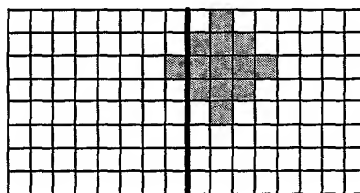column

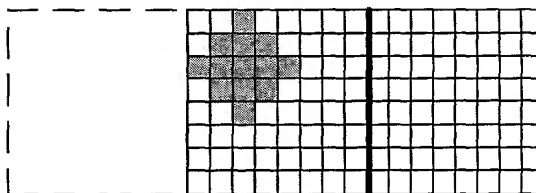Filter centered on 7th
column

Filter centered on 8th
column

Filter centered on 9th
column

Filter centered on 10th
column

Filter centered on 11th
column, left block no longer
required and swapped out

Note that this block is
identical to the 1st block,
identical addressing and
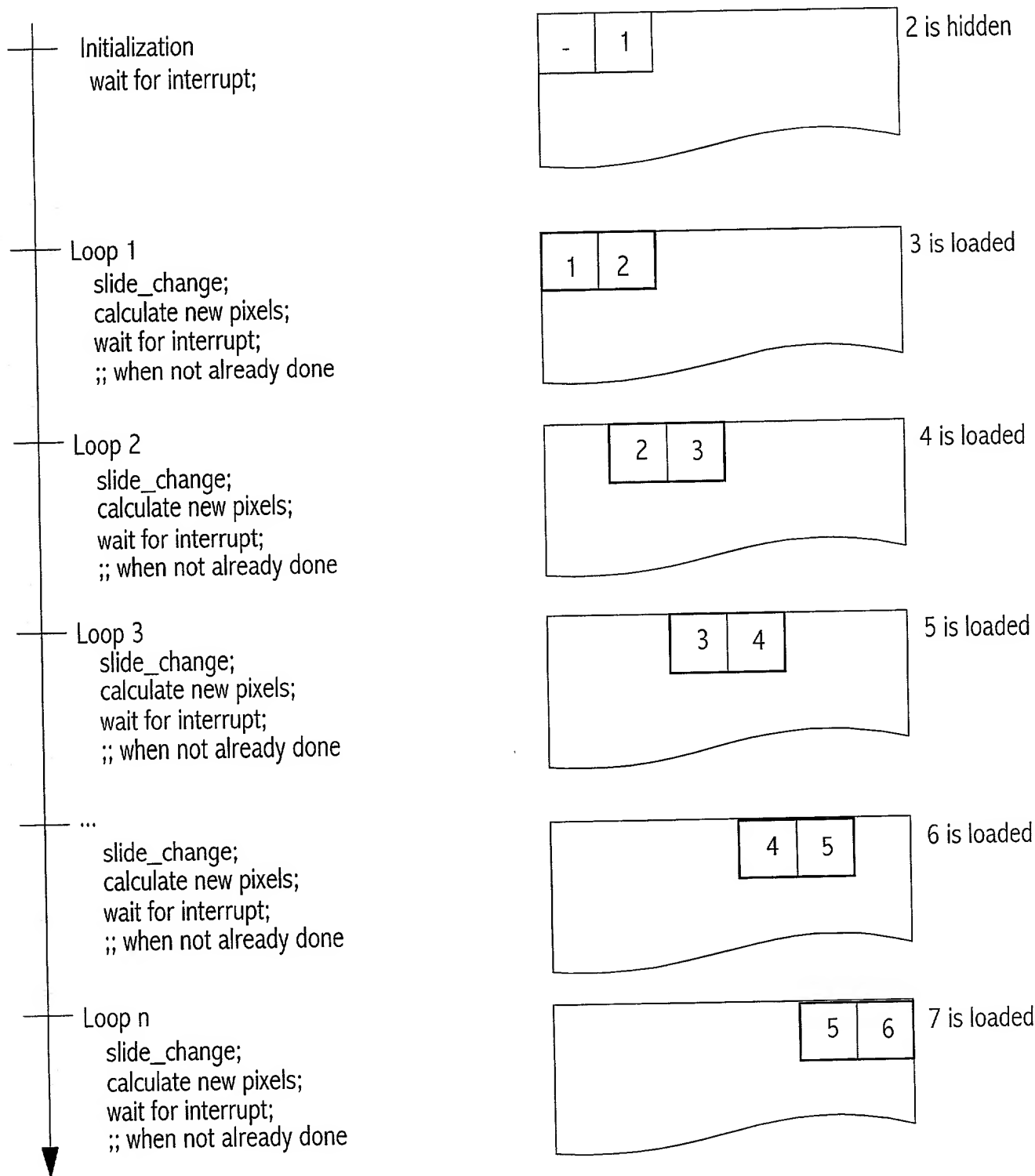thus code can be used.

FIG. 3

Initialization
    wait for interrupt;

Loop 1
    slide_change;
    calculate new pixels;
    wait for interrupt;
    ;; when not already done

Loop 2
    slide_change;
    calculate new pixels;
    wait for interrupt;
    ;; when not already done

Loop 3
    slide_change;
    calculate new pixels;
    wait for interrupt;
    ;; when not already done

...
    slide_change;
    calculate new pixels;
    wait for interrupt;
    ;; when not already done

Loop n
    slide_change;
    calculate new pixels;
    wait for interrupt;
    ;; when not already done

2 is hidden

3 is loaded

4 is loaded

5 is loaded

6 is loaded

7 is loaded

FIG. 4